REMARKS

The Examiner is thanked for the thorough examination of the present application. The Office Action, however, is made final and has rejected all claims. In this response, the applicant has carefully considered the Examiner's opinion and thereby made the amendments to the claims. The applicant has amended claims 1, 3, 7, 8, 9, 10, 19, and 23, and claims 1-24 are currently pending. The features added by these amendments are fully supported by the original application, including FIG. 2A, 2B, 2C, 8, 9A, and 9B, from paragraph [0039] to paragraph [0041], and from paragraph [0067] to paragraph [0079] of the original application; accordingly, the amendments add no new matter to the application. After the amendments, the claims are now believed to be patentably distinct from those cited references and in condition for allowance. Therefore, reconsideration and allowance of the application and currently pending claims are respectfully requested for at least the reasons set forth herein.

Claim Rejections – 35 USC 103

Claims 1, 3, 7, 11-14, 19-21 and 23-24 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over DeKoning et al. (US Patent No. 6,467,023, DeKoning for short hereafter) in view of Randall et al. (US Patent No. 6,530,031, Randall for short hereafter). Claims 2, 4-6, 18 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over DeKoning in view of Randall as applied to claim 1 above, and further in view of TechTarget ("Nonvolatile Storage" TechTarget for short hereafter). Claims 8-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over DeKoning in view of Randall as applied to claim 7 above, and further in view of TechTarget. Claims 15-16 are rejected under 35 U.S.C. 103(a) as being unpatentable over DeKoning in view of Randall as applied to claim 3 above, and further in view of TechTarget. Claim 17 is rejected under 35 U.S.C. 103(a) as being unpatentable over DeKoning in view of Randall in view of TechTarget as applied to claim 2 above, and further in view of Humlicek et al. (US Patent No. 5,822,782).

The applicant respectfully disagrees with the Examiner's opinions because of the following.

Firstly, the applicant will give more detailed explanation for the amended claims 1 and 3 and the cited references in order to let the Examiner understand the initialization progress table and induced consistency initialization progress of the present claimed invention are patentably distinct from the cited references.

In addition, the applicant has also now amended claims 1, 3, 7, 8, 9,10, 19, and 23, and the amended claims 1, 3, 7, 8, 9, 10, 19, and 23 have now included, respectively, features which made the claims more specific. Moreover, claims 2-24 depend from claim 1, either directly or indirectly.

Therefore, the applicant respectfully submits that these rejections should be withdrawn for at least the reasons set forth below.

Response:

Regarding claim 1

The examiner admits that DeKoning et al. (US Patent No. 6,467,023), hereinafter called DeKoning, fails to disclose the initialization progress table recited by the present application. Thus, the Examiner relies on Randall et al. (US Patent No. 6,530,031), hereinafter called Randall, to show the initialization progress table.

The applicant respectfully disagrees with the Examiner's understanding of Randall, and also respectfully disagrees with the obviousness of adding the initialization table based on the teaching of Randall. Please refer to Fig. 3, Fig. 5, and Fig. 6 of Randall as follows.

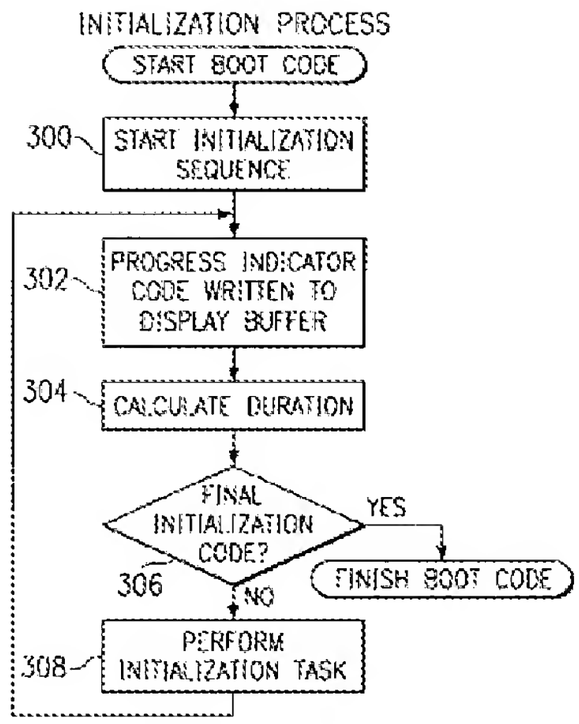


FIG. 3

FIG. 4

PROGRESS INDICATOR CODE 502	TIME STAMP 504	DURATION 506
0100	000 050 000 000	2000
0202	000 450 000 000	900
0301	000 630 000 000	1100
0405	000 850 000 000	
⋮	⋮	⋮

500

INITIALIZATION TIMING TABLE

FIG. 5

SAMPLE CALCULATION

FIG. 6

PROCESSOR FREQUENCY : 200 MHz
CONSTANT: 200,000 TICKS PER MILLISECOND
TIME STAMP FOR TASK 0202 : 450,000,000
TIME STAMP FOR TASK 0301: 630,000,000
DURATION OF TASK 0202: 900 msec

$$\frac{630{,}000{,}000 - 450{,}000{,}000}{200{,}000} = \frac{180{,}000{,}000}{200{,}000} = 900 \text{ msec}$$

In fact, Randall relates to a method to accurately measure, through using the initialization timing table, time durations for self-test of each hardware during boot time of computer.

The initialization timing table in Fig.6 of Randall includes a plurality of rows, and each row is divided into three columns, which are columns 502, 504, and 506 respectively, where the three columns 502, 504, and 506 include progress indicator code, time stamp, and duration respectively.

A boot code, shown in Fig. 3, is executed when the system is started to run, and the boot code contains a sequence of initialization process steps 300, 302, 304, 306, and 308. The step 300 is the operation of START INITIALIZATION SEQUENCE (i.e. loading the first initialization program from ROM). The step 302 is the operation of PROGRESS INDICATOR CODE WRITTEN TO DISPLAY BUFFER (i.e. Each initialization task calls a routine to display a program indicator code.). The step 304 is the operation of CALCULATION DURATION (i.e. The time duration calculation is performed immediately after the progress program is stored.). The step 306 is a selecting operation of FINAL INITIALIZATION CODE (i.e. A detection for the appearance of final code). The step 308 is the operation of PERFORM INITIALIZATION TASK (i.e. The next initialization task is executed if the progress indicator code is not the final code.) (Randall, Column 2, Lines 21-25; Column 4, Lines 20-33).

In addition, the columns 504, 506, and 508 in the initialization table of Randall are respectively utilized to store the progress indicator code for each task, **the time stamp value for indicating the starting time of each task in the progress time base register**, and **the calculated duration of each task in milliseconds** (Randall, Column 4, Lines 35-65; Column 5, Lines 8-13).

In addition, the system of Randall adopts a sequential way to enter each task into the initialization table of Randall **to calculate the elapsed time of each task** (Randall, Column 4, Lines 35-65, 「With reference now to Fig. 4… The time duration is calculated for the initialization task that has been completed, hereafter called task one and stores table entries for the task to be executed next, hereafter called task two. … Each task entered in the initialization table has three associated values: the progress indicator code, the time stamp when the task begins, and calculated time duration in milliseconds. The duration calculation also maintains a data entry point out the table so that data already recorded can be accessed and new data can be entered. The duration of task one is read

from NVRAM (step 406) and used to calculate the elapsed time in milliseconds (step 408) for task one. ... 」).

In conclusion, the initialization timing table in Fig. 6 of Randall is intended to calculate the elapsed time of each task.

The disclosure of Randall is summarized as follows. Randall discloses **a boot code task duration calculation method** used in **a boot sequence** of **a computer**, using a **table** (initialization timing table) to calculate the elapsed time of each boot sequence tasks.

The structure of the initialization timing table comprises a plurality of records (rows), each including three fields (columns), which are columns 502 including progress indicator code, 504 including time stamp, and 506 including duration.

In other words, each record of Randall's table (initialization timing table) stores a boot sequence task identifier number (the indicator code) and the task operating time-related information (time stamp) and duration corresponding thereto calculated from the time-related information.

This initialization timing table is provided by Randall to solve the problem of no easy and precise way to estimate the time interval used by each booting sequence tasks of a computer while booting in the prior arts. **Obviously, the "initialization" here in Randall means the "computer booting." No region of PSDs needs to be initialized in Randall. No initialization region state is recorded in a table in Randall.**

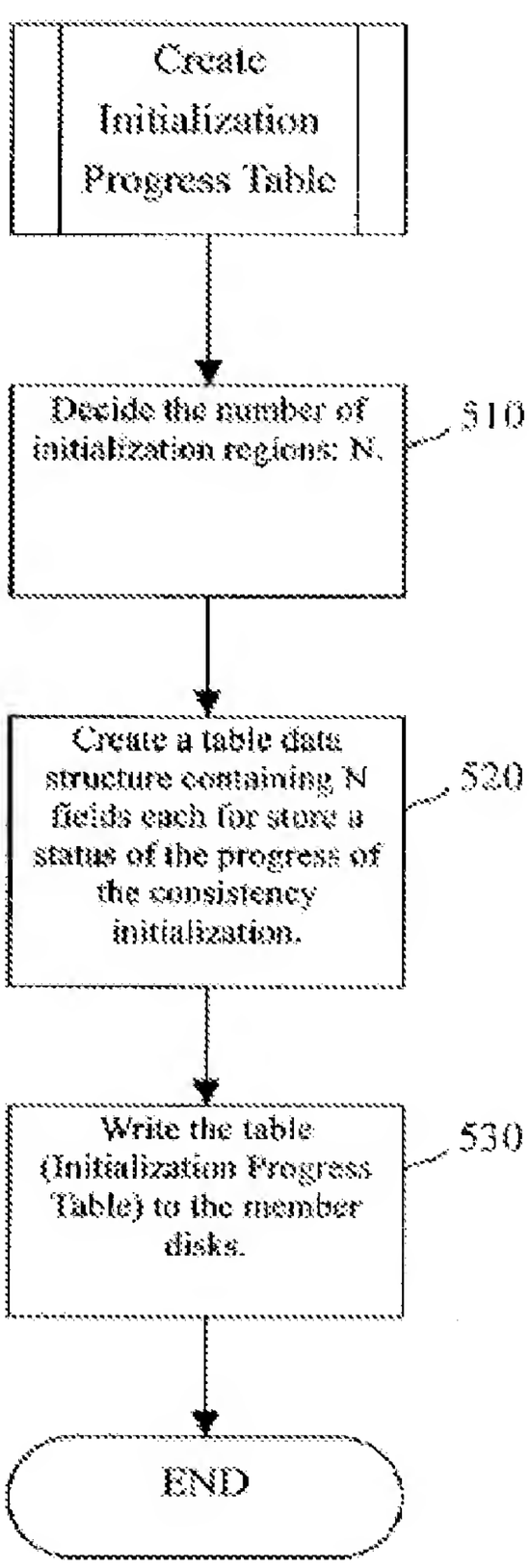
In contrast, different from field invention of Dekoning, the present invention relates to initialization of physical storage devices (PSD) of a RAID subsystem.

The claimed invention is **a physical storage device (PSD) consistency initialization method** used **in a RAID subsystem**, using **an initialization progress table** which comprises a plurality of **fields, each corresponding to a region of the PSDs to record the initialization state of the PSD regions.**

A RAID subsystem usually comprises a plurality of PSDs (physical storage devices, such as hard disk drives), and a storage controller for connecting the PSDs to a computer to provide data storage space to the computer.

Before a RAID subsystem is used for the first time, the PSDs must be initialized by using the initialization progress table. The consistency initialization of the present invention is performed on the PSDs by the storage controller to make the storage medium in the PSDs in a region-by-region basis. **Apparently, the "consistency initialization" here in the claimed invention means the "data in the regions of PSDs in a RAID**

Figure 5A



subsystem is made consistent before the regions are used."

Obviously, the consistency initialization is irrelevant to a computer booting

sequence.

In addition, independent claim 1 of the present invention claims in part "the initialization progress table including a plurality of fields, each of which is used to record whether a regional initialization is performed on an initialization region," which is described in detail as the following:

According to [0048] of the present invention, "please refer to FIG. 5A, which is a flow chart of an embodiment for creating an initialization progress table according to the present invention. As shown in FIG. 5A, a value N is first determined as the number of regions the member disks of the RAID are to be divided into (510), where each region is a basic unit for performing consistency initialization. Then an initialization progress table is created having N fields to record the initialization state of each initialization region (520). After the initialization progress table is written to each member disks (530), and the process of initialization progress table creation is completed. The above-mentioned basic units are called initialization regions and the initialization on each initialization region is called regional initialization. The present invention consistency initialization is performed by performing regional initialization on each initialization region. When the regional initialization on all initialization regions is completed and the completion state is stored in the reserved space of the member disks, the consistency initialization is completed."

The differences between the claimed invention and the Randall reference are provided in the following Comparison Table.

Comparison Table

Differences	Present claimed invention	Randall reference
The device using the method	**RAID subsystem**, which is a storage subsystem **comprising** a storage controller and a **plurality of PSDs each having multiple regions to be initialized**.	**Computer**
The invented method	**Consistency initialization method for PSD regions**	Boot task **duration calculation method**

Use timing	When the RAID subsystem is used for the first time, **to initialize the regions of the PSDs.**	When the computer is **booting.**
Table structure	Include multiple fields, each for a region of the PSDs, **to record initialization state of each of the regions.**	Include multiple records (rows) each for a boot sequence task, each record has multiple fields (column) to store the indicator code of a task, time stamp, and duration. **No initialization region state is recorded.**
Purpose of table used	Use a table (initialization progress table) **to store the initialization state of each of the regions of the PSDs**.	Use a table (initialization timing table) to store each boot sequence task so as **to calculate the elapsed time of each task.**
Regions in **PSDs to be initialized?**	**Yes.** A storage subsystem has **many PSDs to perform thereon consistency initialization.**	**No!** A computer has **no PSDs to be initialized in boot sequence**. **No PSD consistency initialization thing when booting.**
Meaning of "initialization"	**"initialization" means "consistency initialization," which refers to a situation that data in the regions of PSDs in a RAID subsystem is made consistent before the regions are used.**	**"initialization" refers to computer booting, which does not relate to "consistency initialization" of the regions of PSDs.**
Disclose claimed features?	**Yes. "the initialization progress table including a plurality of fields, each of which is used to record whether a regional initialization is performed on**	**No.** Randall does not disclose the claimed features as the left.

	an initialization region"	

From above paragraphs, explanations, figures and comparisons, it can be concluded that, the initialization timing table of Randall is entirely different from the initialization progress table of the present invention and does not disclose the claimed invention in the amended claim 1.

Moreover, in addition to the fact that the initialization timing table in Fig. 6 of Randall is different from the initialization progress table of the present invention, the initialization progress table of the present application discloses three features, which Randall fails to disclose, as mentioned below.

(1) The initialization progress table of the present application relates to the record for the states of initialization region. In other words, the initialization progress table of the present application is created having N fields to record the initialization state of each initialization region when the RAID is built and is immediately divided into N initialization regions, where each region of the N initialization regions is a basic unit for performing consistency initialization.

(2) The initialization progress table of the present application relates to the record for the number of the initialization regions. As mentioned above, the number of the initialization regions is determined after the RAID is built and is divided into N initialization regions. In other words, the system can realize how many initialization regions should be initialized at the beginning.

(3) The system can initialize, according to the initialization progress table of the present invention, the N initialization regions through using the consecutive consistency initialization and the induced consistency initialization in parallel because the initialization progress table of the present application can show which initialization regions have been initialized and which initialization region have not yet been initialized.

Furthermore, according to MPEP, 2106 II C, "Finally, when evaluating the scope

of a claim, every limitation in the claim must be considered. USPTO personnel may not dissect a claimed invention into discrete elements and then evaluate the elements in isolation. Instead, the claim as a whole must be considered." However, it is believed that some limitations in independent claim 1 of the present invention have never been actually considered, such as "the initialization progress table including a plurality of fields, each of which is used to record whether a regional initialization is performed on an initialization region" in independent claim 1 of the present invention, and that claim 1 of the present invention as a whole has never been considered.

For aforesaid reasons and MPEP paragraphs, the applicant of the present invention hereby begs the Examiner's favor in reconsidering every limitation in the claim1 of the present invention.

Regarding claim 3

The Examiner deems that DeKoning has disclosed the features of the induced consistency initialization claimed in claim 3 of the present application.

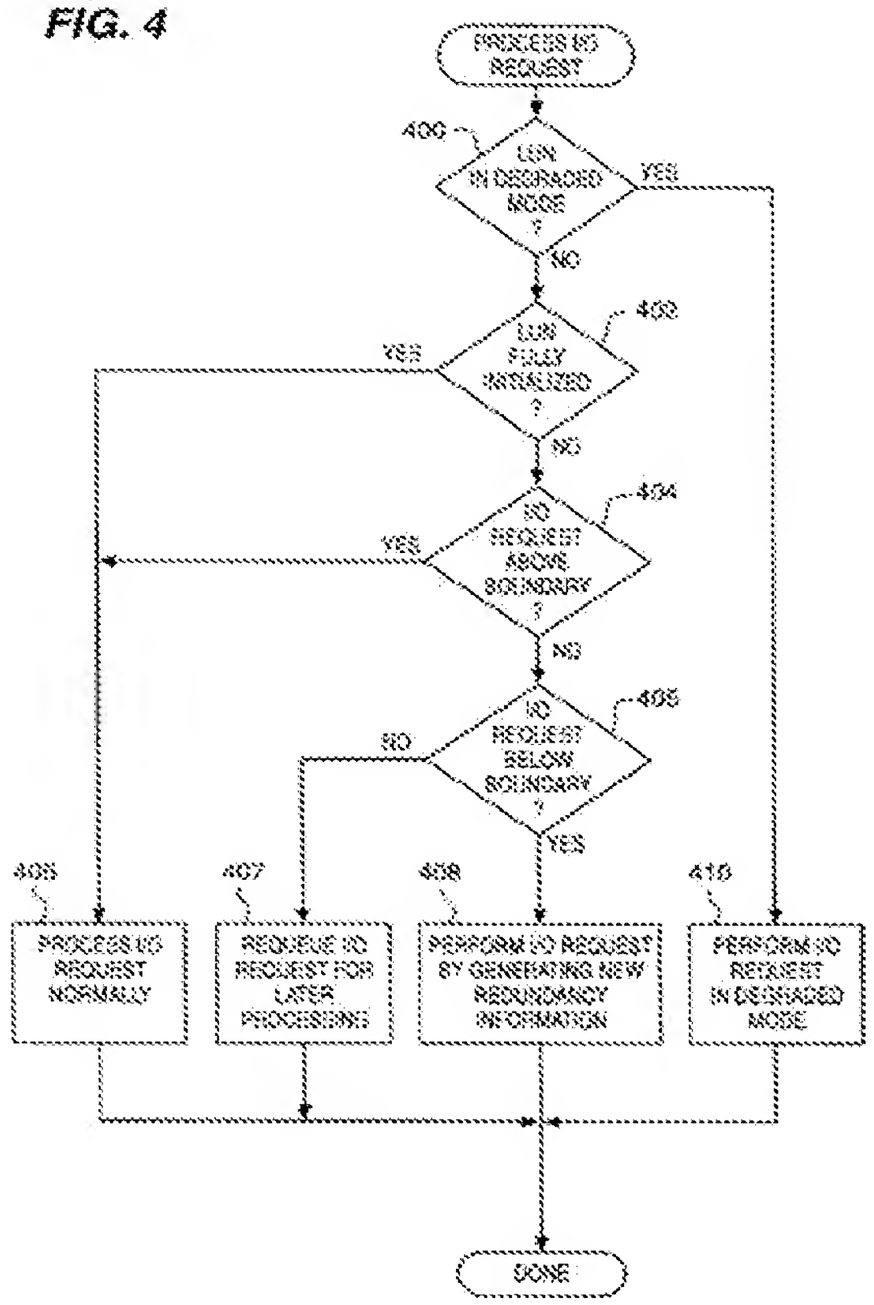
The applicant respectfully disagrees with the Examiner's understandings of DeKoning and of the induced consistency initialization of the present invention, and wants to explicitly explain the differences between the invention of DeKoning and the induced consistency initialization of the present invention, and to explicitly explain why element 407 and element 408 of Dekoning have never disclosed the induced consistency initialization of the present invention as follows:

According to DeKoning, the LUN is divided into a plurality of portions, and the portions of the LUN are initialized sequentially one after another from top to bottom. Therefore, **the initialization boundary is recorded by using a local variable CURRENT and the state information thereof is saved as checkpoint data**. Before the completion of the initialization, if the initialization is interrupted due to power loss or something else, the initialization progress can be resumed after the interruption cause disappears, by using the checkpoint data (DeKoning, Column 8, Lines 1-6, 「Element 307 then saves state information regarding the progress of the LUN initialization. This saved

data (also referred to herein as checkpoint data) is used later if the initialization process is resumed after being interrupted (i.e., by a power loss in the storage system).」).

Further, the applicant of the present invention notes that **the system generates a new redundancy data to make the initialization progress being performed when the system accepts a host I/O request attempting to access an un-initialized LUN,** i.e. the LUN is below the boundary level (DeKoning, Column 3, Lines 56-60, 「Host I/O requests for writing of data below this boundary level (i.e., in an area of the LUN for which redundancy information has not yet been initialized) are performed in a manner that assures valid redundancy data is generated and written.」) (DeKoning, Column 9, Lines 35-51, 「Those skilled in the art will recognize that the processing of element 408 to generate new redundancy information is performed in accordance with the particular RAID management level associated with the LUN. For example, where the LUN is managed in accordance with RAID level 1 mirroring, generation of new redundancy information entails writing of the user requested data and duplicating the data in the associated mirror portion of the LUN. By contrast, where the LUN is managed in accordance with RAID level 5 techniques, processing of element 408 entails generation of new redundancy information by Exclusive OR parity operations including the data modified by the I/O write request and any associated old data remaining on the LUN in the related stripe. It all such cases, operation of element 408 leaves the redundancy information associated with the I/O write request in a consistent state such that it maybe relied upon in processing of subsequent read or write requests.」).

FIG. 4



Moreover, according to element 407 of Fig. 4 shown as below of DeKoning. The element 407 of DeKoning describes that if the region accessed by a I/O request is neither fully above nor fully below the present boundary but spans the present boundary, it will generates a queue to handle the I/O request for later process. (DeKoning, Column 9, Lines 18-21, 「If the data to be accessed is neither fully above nor fully below the present boundary but rather spans the boundary, element 407 is operable to requeue the I/O request for later processing.」). Therefore, if un-initialized areas of the LUN, to which a host I/O attempts to make access, are on the lowest location of the disk drive, then the un-initialized areas on the lowest location of the disk drive will not be initialized and will not be accessed by the host I/O until the un-initialized areas of the LUN, to which the host I/O attempts to make access, have been already initialized, which no doubt makes the element 407 of DeKoning time-consuming, because element 407 of DeKoning has to wait

for completion of initializing the un-initialized areas on the lowest location of the disk drive or ordinary memory space, and then after completion of initializing these un-initialized areas, I/O request is perform through making data access to these initialized areas on the lowest location of the disk drive or ordinary memory space.

On the contrary, according to claim 3 of the present invention, **when an initialization region, to which a host I/O request attempts to make access, is an un-initialized region, the induced consistency initialization of the present application is activated to first initialize the initialization region that is associated with the host I/O request in order to let the initialization region then be accessed by the host I/O request**. In order to explain in detail, please refer to the paragraphs [0039], [0054], [0064], and [0079] of the present application respectively, which recite that "Induced consistency initialization is the consistency initialization that is induced by the access of an I/O to the RAID on a region which is not initialized, after the RAID creation is completed (280A)", "when the host entity accesses a RAID and a associated RAID controller receives a host I/O command and parses the command to access a initialization region corresponding to the host I/O command, if the initialization region is not initialized, the regional initialization is induced to be performed due to the host I/O command.", "Moreover, the priority of the regional initialization induced by the I/O command can be raised in the initialization.", and "The point is, according to the induced consistency initialization of the present invention, when a RAID is completed with the RAID CREATE procedure, the RAID is allowed to be accessed by I/O and when a region accessed by an I/O is not performed regional consistency initialization, it will start regional consistency initialization on the associated region.".

Moreover, **if a Host I/O request is accepted during the initialization progress, the induced consistency initialization is activated according to the information of the initialization progress table to initialize the initialization region, to which the host I/O request tries to make access to**.

From above paragraphs, explanations, figures and comparisons, it can be concluded

as follows:

(1) Elements 407 and 408 of Dekoning have nothing to do with claim 3 of the present invention at all. (Please refer to aforesaid paragraphs)

(2) On the contrary, Claim 3 of the present invention first initializes these un-initialized regions that are associated with I/O request, which makes claim 3 of the present invention faster and more efficient than elements 407 and 408 of Dekoning.

(3) According to the above-mentioned reasons, it is believed that DeKoning fails to disclose the features of induced consistency initialization claimed in claim 3 of the present application. Accordingly, the applicant asserts that claim 3 and the dependent claims thereof are allowable over these cited references.

Regarding claims 2 and 24

Claims 2-24 pertain to claim 1 directly or indirectly and as such are also considered to be allowable if claim 1 is found allowable. In addition, each of these claims has other features that make them additionally allowable.

Conclusion

For the reasons as described above, the applicant believes that pending claims 1-24 are allowable over the cited references. Withdrawal of the rejections of claims is respectfully requested. Should the Examiner feels that further discussion of the application and the amendment is conducive to prosecution and allowance thereof, please do not hesitate to contact the undersigned at the address and telephone listed below.

Sincerely yours,

/Winston Hsu/ Date: 11/27/2009

Winston Hsu, Patent Agent No. 41,526
P.O. BOX 506, Merrifield, VA 22116, U.S.A.
Voice Mail: 302-729-1562
Facsimile: 806-498-6673
e-mail : winstonhsu@naipo.com

Note: Please leave a message in my voice mail if you need to talk to me. (The time in D.C. is 13 hours behind the Taiwan time, i.e. 9 AM in D.C. = 10 PM in Taiwan.)